UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/774,584	02/10/2004	Yao-Ching Stephen Chen	SVL920030107US1	9752

45727 7590 09/19/2006

LACASSE & ASSOCIATES, LLC
1725 DUKE STREET, SUITE 650
ALEXANDRIA, VA 22314

EXAMINER	
SYED, FARHAN M	

ART UNIT	PAPER NUMBER
2165	

DATE MAILED: 09/19/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/774,584	CHEN ET AL.	
	Examiner	Art Unit	
	Farhan M. Syed	2165	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 February 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *10 February 2004* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *02102004*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-26 are pending.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1-26 are provisionally rejected on the ground of nonstatutory double patenting over claims 1-37 of copending Application No. 10/774,594. This is a provisional double patenting rejection since the conflicting claims have not yet been patented.

Claims 1-37 of 10/774,594 reference recites all the elements of claims 1-26 of the instant application 10/774,584. The scope of 10/774,594 as recited in the claims 1-37 is a method for markup language schema validation, comprising the steps of: (a)

loading a markup language document into a runtime validation engine, wherein the runtime validation engine comprises a mark up language schema validation parser; (b) loading an annotated automaton encoding for a mark up language schema definition into the markup language schema validation parser; and (c) validating the markup language document against the markup language schema definition by the markup language schema validation parser utilizing the annotated automaton encoding. The scope of 10/774,584 as recited in claims 1-26 is a method for compiling a structured document schema into type annotation records comprising steps of: a. building a type hierarchy ordered tree from a structured document schema from type record wherein each of said type records contains typing tuples, b. creating a typing set containing said typing tuples in said type hierarchy ordered tree, c. creating an ambiguity typing sequence for said typing tuples sharing a common first field and having a unique second field, d. arranging said ambiguity typing sequence based on an offset number assigned to a third field of each of said typing tuples in said ambiguity typing sequence, e. extracting a second field from each of said typing tuples accorded to sorted order of said ambiguity typing sequences, and creating a type indexing data structure populated with said extracted second field to map each type name to a type. The difference between the inventions is that the instant application 10/774,584 is a method for compiling a structured document schema into type annotation records and application no. 10/774,594 is a method for markup language schema validation. The reason why a person of ordinary skill in the art would conclude that the invention defined in claims 1-26 of application no. 10/774,584 is that it would have been an obvious variation of the

invention defined in claims 1-37 of application no. 10/774,594. In light of these factual determinations, Based on the ordinary skill pertinent in the art and is as such obvious and as such anticipates claims 1-26 of the instant application.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Drawings

4. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference character(s) not mentioned in the description: Figure 2, items 212, 214, and 220; Figure 3, items 300, 302, 308, 310, 314, 318, and 320; Figure 4, items 402-416; and Figure 6, item 604. Corrected drawing sheets in compliance with 37 CFR 1.121(d), or amendment to the specification to add the reference character(s) in the description in compliance with 37 CFR 1.121(b) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

2. Claims 1-26 are rejected under 35 U.S.C. 102(e) as being anticipated by a non-patent literature titled "An Efficient XML Schema Typing System" by Wang, Ning and et al., pages 1-21, published 18 Nov. 2003 (and known hereinafter as Wang).

The applied reference has a common inventor with the instant application. Based upon the earlier effective U.S. filing date of the reference, it constitutes prior art under 35 U.S.C. 102(e). This rejection under 35 U.S.C. 102(e) might be overcome either by a showing under 37 CFR 1.132 that any invention disclosed but not claimed in the reference was derived from the inventor of this application and is thus not the invention "by another," or by an appropriate showing under 37 CFR 1.131.

As per claims 1 and 25, Wang teaches a method for compiling a structured document schema into type annotation records (i.e. *"Figure 1 shows the architecture of the XML typing module implementation. The main components are a Generic XML Parser, a Scanner Pool, the XML Typing engine, and an XML Schema Compiler. The XML Schema compiler compiles an XML Schema to automata encoding storage format."*)(Section 1.1) comprising steps of: a. building a

type hierarchy ordered tree from a structured document schema from type record wherein each of said type records contains typing tuples (i.e. *"Traditionally, XML Schema validation is done using a general-purpose XML Schema validation parser, which parses an XML document and XML Schema definition into internal tree automaton formats, then traverses the document tree and checks it against the internal tree automaton."*)(Section 1), b. creating a typing set containing said typing tuples in said type hierarchy ordered tree (i.e. *"Traditionally, XML Schema validation is done using a general-purpose XML Schema validation parser, which parses an XML document and XML Schema definition into internal tree automaton formats, then traverses the document tree and checks it against the internal tree automaton."*)(Section 1), c. creating an ambiguity typing sequence for said typing tuples sharing a common first field and having a unique second field (i.e. *"If a terminal has ambiguous types then the type# of its terminal dictionary entry points to one of its type. The remaining types are stored contiguously in typearray after its type#."*)(Section 3.5.1), d. arranging said ambiguity typing sequence based on an offset number assigned to a third field of each of said typing tuples in said ambiguity typing sequence (i.e. *"A field offset is assigned to all types in Tc. The remaining types will be addressed by type# + offset. " "The order [sic] of entries in the typearray are correlated with the offset of each type. Before we present the algorithm which determines the order of entries in the typearray and the offset of each type, we first introduce a relation called compete.* ***Definition 5 (compete)*** *8 Tc:x; Tc:y 2 Tc. Tc:x competes with Tc:y if and only if a non-terminal Xi in Tc:x binds a terminal al to a type Ti and while a non-terminal Xj in Tc:y binds the same terminal al to a different type Tj. This relation is denoted by Tc:x v Tc:y. "*)(Section 3.5.1, Section 3.5.2), e. extracting a second field from each of said typing tuples accorded to sorted order of said ambiguity typing sequences, and creating a type indexing data structure populated with said extracted second field to map each type name to a type (i.e. *"For all edges (Tx; Ty) in a type reference graph, reverse edges (Ty; Tx) are*

added into the type reference graph such that given a type we can find the content model of that type. The improved data structure with the extended terminal dictionary and reverse edges is illustrated in Figure 6." "For example, the following is a valid assignment of offsets to types in Figure 7. Tx with offset=0, Ty with offset=1, Tz with offset=2, Tv with offset=3, Tu with offset=4, Tw with offset=5. Ts can be assigned to any empty slot in the typearray since it doesn't compete with any other types. Once we have offset for each type, we need to assign a unique type# to all terminals."(Section 3.5.1, Section 3.5.2).

As per claim 2, Wang teaches a method for the compilation of a structured document schema, wherein said structured document schema is an XML document schema (i.e. *"Type validation, in particular XML Schema validation, checks if XML data conforms to its type definition. We need to be able to check any type in T. Recall that T = Ts [Tc. Because Ts and Tc are sets of regular expressions over different domains, they need to be processed separately."*)(Section 3.4).

As per claim 3, Wang teaches a method wherein said typing tuples in said typing set are sorted to create said ambiguity typing sequence (Figure 2).

As per claim 4, Wang teaches a method wherein said arranging step is further comprised of: collecting each third field of said typing tuples and sorting said typing tuples in said ambiguity sequence with respect to third field of said typing tuple (i.e. *"For example, the following is a valid assignment of offsets to types in Figure 7. Tx with offset=0, Ty with offset=1, Tz with offset=2, Tv with offset=3, Tu with offset=4, Tw with offset=5."*)(Section 3.5.2).

As per claim 5, Wang teaches a method wherein a typing tuple is comprised of an element type name in said first field, a type identifier in said second field, and a parent element name in said third field (Figure 2).

As per claim 6, Wang teaches a method, wherein said name in said first field is used in said sorting step to alphabetically sort said typing tuples in typing set (i.e. *"**Definition 2 (Grammar of XML Type)** The grammar of XML type Tc:xsd is a 5-tuple G = (N, I, S 0, T, P), where:1. N is finite set of non-terminals. 2. I is finite set of terminals (XML element names). 3. T = Ts [Tc, Ts is a finite set of regular expression Ts:x over the input alphabet. Tc is a finite set of regular expression Tc:x over N. Each regular expression Tx is assigned a unique type name. Two types are equivalent if and only if both types have the same type name. 4. S 0 is the start symbol and S 0 ! TTc:xsd , where Tc:xsd 2 Tc and Tc:xsd denotes the type defined by an XML Schema. 5. P is a finite set of production rules of the form Xi ! aiTx, where Xi 2 N; ai 2 I and Tx 2 T and Xi has only one production rule and each production rule binds one terminal ai to a type Tx."*)(Section 3.1).

As per claim 7, Wang teaches a method wherein said name is one of: a type name, element name, or attribute name; and said type identifier is one of: a type, element, or attribute (Figure 2).

As per claim 8, Wang teaches a method wherein said third field is empty if said parent element name corresponds to a global element type (Figure 2).

As per claim 9, Wang teaches a method wherein a typing set is comprised of distinct typing tuples, wherein two typing tuples are distinct if either said first fields of

both of said typing tuples are different or said second fields of both of said typing tuples are different (i.e. *"Type validation, in particular XML Schema validation, checks if XML data conforms to its type definition. We need to be able to check any type in T. Recall that T = Ts [Tc. Because Ts and Tc are sets of regular expressions over different domains, they need to be processed separately."*)(Section 3.4).

As per claim 10, Wang teaches a method wherein said offset in said arranging is the position of said ambiguity type in an ambiguity typing sequence (i.e. *"XML Schema [17] allows both named type declaration and anonymous type declaration. A named type is unique in its namespace. An anonymous type declaration is required to be assigned a unique name in its namespace in the data model used by XQuery 1.0 and XPath 2.0 [18]."*)(Figure 2; Section 2.1, section 3).

As per claim 11, Wang teaches a method wherein said type indexing data structure can be any one of: a hash table, a binary tree, and a B+ tree (i.e. *"Based on the above observation, the terminal dictionary is extended. Each entry in the terminal dictionary contains four fields namely terminal, tok, ntypes, and type#. The ntypes field is the number of types bound to a terminal. type# is an index into a typearray."*)(Section 3.5.1).

As per claim 12, Wang teaches a method wherein said type indexing data structure is comprised of a column indicating ambiguity type for each of said type names and a column indicating offset (i.e. *"The order of entries in the typearray are correlated with the offset of each type. Before we present the algorithm which determines the order of entries in the typearray and the offset of each type, we first introduce a relation called compete."*)(Section 3.5.2).

As per claim 13, Wang teaches a method for a database engine to perform type annotation of structured documents or structured document fragments in the absence of full schema validation (i.e. *"The validation algorithm can validate both XML documents and XML document fragments. If we need to validate XML data against an XML Schema, then we can invoke validate(T; XML stream) with T = Tc:xsd . If the XML data is a well-formed fragment, then T is the assigned type of the XML fragment." "Type-annotation can be separated from validation. The user has the choice of doing both validation and type annotation, or type annotation only."*)(Section 3.4, Section 1.2), comprising steps of: a. building a type annotation data structure comprising a structured document type hierarchy, a type indexing data structure, and a type array (i.e. *"In this section we define a concise model to capture the essence of XML Schema. Starting with an XML Schema example expressed in the form of the concise model, we can map the model to a data structure step by step. Type annotation and validation are addressed, and querying type using of an enhanced data structure and algorithm is also discussed." "In many cases, if we know that XML data is well-typed, we can annotate XML data with type information without re-validating the XML data. The function typing : Tc T I ! T provides sufficient information for type annotation. The type annotation algorithm is given below." "Based on the above observation, the terminal dictionary is extended. Each entry in the terminal dictionary contains four fields namely terminal, tok, ntypes, and type#. The ntypes field is the number of types bound to a terminal. type# is an index into a typearray."*)(Section 3; 3.3, 3.5.1), b. mapping a type name string to each element type in said structured document type hierarchy, and annotating a structured document or fragment using type annotation records obtained from said type annotation data structure and said type name mapping (i.e. *"**Basic validation** against XML Schema and incremental validation [13] after update are necessary for an XML database system. To provide strong typing, XML data will need type annotation with or without validation. Accurate and highly efficient access to XML Schema and type information is the basis for all the areas of XML Schema-aware applications to take advantage of strong typing. Traditionally, XML Schema*

validation is done using a general-purpose XML Schema validation parser, which parses an XML document and XML Schema definition into internal tree automaton formats, then traverses the document tree and checks it against the internal tree automaton.")(Section 1).

As per claims 14 and 26, Wang teaches a method for a database engine to perform type annotation, wherein said mapping step further comprises steps of: a. loading said type annotation data structure into a runtime validation engine (i.e. *"This typing system can be used as part of the runtime environment for XML processing languages that adopt XML Schema as the type system."*)(Abstract), b. creating an empty offset stack data structure (i.e. *"A field offset is assigned to all types in Tc. The remaining types will be addressed by type# + offset."*)(Section 3.5.1), c. pushing record containing a value of zero onto said offset stack (i.e. *"The first type is the type Tc:name0 pointed by its type#. The second type, Ts:string, is the non-empty slot in typearray following the type#."*)(Section 3.5.2), d. using a token from an XML document or document fragment to key a search on a type indexing data structure to determine an index for said token (i.e. *"The validation algorithm can validate both XML documents and XML document fragments. If we need to validate XML data against an XML Schema, then we can invoke validate(T; XML stream) with T = Tc:xsd . If the XML data is a well-formed fragment, then T is the assigned type of the XML fragment."*)(Section 3.4), e. incrementing said index by value in topmost record of offset stack if said token is indicated to be of ambiguous type, and indicating element type in a type array at said index location (i.e. *"XML Schema [17] allows both named type declaration and anonymous type declaration. A named type is unique in its namespace. An anonymous type declaration is required to be assigned a unique name in its namespace in the data model used by XQuery 1.0 and XPath 2.0 [18]."*)(Section 2.1).

As per claim 15, Wang teaches a method for a database engine to perform type annotation, herein said type is an XML type (i.e. *"XML Schema uses a named type system, making type checking simple."*)(Section 2.1).

As per claim 16, Wang teaches a method for a database engine to perform type annotation, wherein said record is a type annotation record (i.e. *"Type annotation and validation are addressed, and querying type using of an enhanced data structure and algorithm is also discussed."*)(Section 3).

As per claim 17, Wang teaches a method for a database engine to perform type annotation, wherein said method supports defaults, "any" type, and "xsi:nil=`true`" attribute (Figure 1).

As per claim 18, Wang teaches a method for a database engine to perform type annotation, wherein attribute defaults are supported by associating attribute types with element types in said type annotation records (Figure 1).

As per claim 19, Wang teaches a method for a database engine to perform type annotation, wherein a type is annotated with "any" type if an index is not located for said token in said searching step (i.e. *"Based on the above observation, the terminal dictionary is extended. Each entry in the terminal dictionary contains four fields namely terminal, tok, ntypes, and type#. The ntypes field is the number of types bound to a terminal. type# is an index into a typearray."*)(Section 3.5.1).

As per claim 20, Wang teaches a method for a database engine to perform type annotation, wherein said method is not performed if an "xsi:nil=`true` attribute is encountered (Figure 1).

As per claim 21, Wang teaches a method for a database engine to perform type annotation, wherein said token comprises any of: a start tag and element name; a start tag, element name, and type name; an attribute type and attribute name; or an end tag (i.e. *"The first type is the typeTc:name0 pointed by its type#. The second type, Ts:string, is the non-empty slot in typearray following the type#. Then we start from the node of Tc:name0, Ts:string in the type reference graph respectively traversing back to their referrer. Until Tc:xsd , then both Tc:name0 and Ts:string are qualified types of (Tc:xsd ; ==name). Assume Tc:xsd doesn't not appear on the back traversing path start from Ts:string, then Ts:string is not a qualified type."*)(Section 3.5.1).

As per claim 22, Wang teaches a method for a database engine to perform type annotation, wherein said ambiguous type of said token is determined by a consultation of said typing array (i.e. *"The oder of entries in the typearray are correlated with the offset of each type. Before we present the algorithm which determines the order of entries in the typearray and the offset of each type, we first introduce a relation called compete." "The annotation provides relationships among types and finite automata provide efficient validation mechanism. Together, the proposed techniques address the needs for efficient validation, type annotation and quick access to type information contained in XML Schemata."*)(Section 5).

As per claim 23, Wang teaches a method for a database engine to perform type annotation, wherein a record is pushed onto said offset stack if said token is either a start tag and element name; or a start tag, element name, and type name (i.e. *"The first type is the typeTc:name0 pointed by its type#. The second type, Ts:string, is the non-empty slot in typearray following the type#. Then we start from the node of Tc:name0, Ts:string in the type reference graph respectively traversing back to their referrer. Until Tc:xsd , then both Tc:name0 and Ts:string are qualified types of (Tc:xsd ; ==name). Assume Tc:xsd doesn't not appear on the back traversing path start from Ts:string, then Ts:string is not a qualified type."*)(Section 3.5.1).

As per claim 24, Wang teaches a method for a database engine to perform type annotation, wherein if said token is an end tag; a topmost record of said offset stack is removed (i.e. *"In many cases, if we know that XML data is well-typed, we can annotate XML data with type information without re-validating the XML data. The function typing : Tc T I ! T provides sufficient information for type annotation. The type annotation algorithm is given below." A field offset is assigned to all types in Tc. The remaining types will be addressed by type# + offset."*)(Section 3.3).

Contact Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Farhan M. Syed whose telephone number is 571-272-7191. The examiner can normally be reached on 8:30AM-5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey Gaffin can be reached on 571-272-4146. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

FMS

JEFFREY GAFFIN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100